

پروژه سوم علوم اعصاب محاسباتی

- صورت پروژه در این آدرس قابل مشاهده است.
- با توجه به دشواری حفظ ساختار صورت پروژه در گزارش، آن ساختار نادیده گرفته شده و مطالب با ساختاری مناسب برای دنبال کردن نمودارها و مطالب منظم شده‌اند؛ با اینحال تمام مطالب خواسته شده در صورت پروژه، در این گزارش پوشانده شده‌اند.
- در فازهای قبل به بررسی پارامترهای نورون‌ها پرداختیم. بنابراین در این فاز این پارامترها را مورد بررسی قرار نخواهیم داد و بر روی پارامترهای اتصالات تمرکز خواهیم کرد.
- در این فاز توانایی پنهان کردن کدها در ژوپیتر نوت‌بوک را پیدا نکردیم. این اصلاح در فازهای بعدی انجام خواهد شد.
- در این فاز از جامعه ناهمگن استفاده خواهیم کرد.

0. فهرست مطالب

1. شدت جریان ورودی
2. نوع و تعداد اتصالات
3. مقداردهی وزن‌های اتصالات
4. تغییرات ثابت‌زمانی τ_s

```
In [1]: from cnsproject.network.neural_populations import LIFPopulation
from cnsproject.network.connections import *
from cnsproject.network.monitors import Monitor
from cnsproject.plotting.plotting import Plotter
from cnsproject.utils import *
import matplotlib.pyplot as plt
import torch
import warnings
warnings.filterwarnings("ignore")
```

بجز ابزار شبیه‌سازی (که import شده‌اند)، تابع پایین در این تمرین خاص، برای شبیه‌سازی و مقایسه نورون‌ها در کنار هم به ما کمک خواهد کرد. همچنین در این تمرین، هر شبیه‌سازی را به مدت 200ms ادامه خواهیم داد.

```
In [22]: time=250 #ms
def population_behaviour(population_cls, I, p, time=time, postfix='', dt=1.,
    is_excitatory = torch.ones(100, dtype=bool)
    is_excitatory[:20] = False
    population = population_cls((100,), is_excitatory=is_excitatory, dt=dt, *
    connection = SimpleConnection(pre=population, post=population, dt=dt, **a
    monitor = Monitor(population, state_variables=["s"], time=time)
    population.refractory_and_reset()
    def func(I):
        population.forward(I+connection.I)
        connection.forward(population.s)
    monitor.simulate(func, {'I': I})
    p.monitor = monitor
    data = p.population_activity_raster('s'+postfix, x_vis=False, y_label='sp
    p.population_activity('a'+postfix, raster_data=data, x_vis=False, y_label
    if cd:
        p.current_dynamic('i'+postfix, I=I, y_label="I (mA)", repeat_till=tim
```

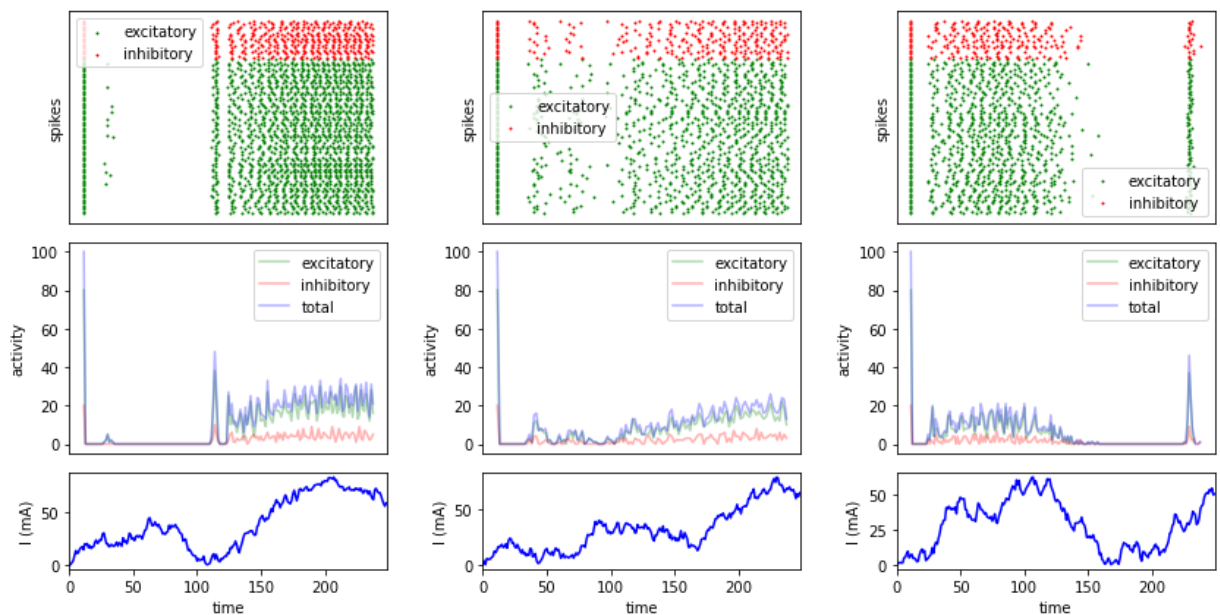
1. شدت جریان ورودی

در نمودارهای زیر، سه نمونه جریان تصادفی تولید و رفتار جمعیت رسم شده است:

```
In [23]: plt.figure(figsize=(14,7))
p = Plotter([
    ['s1','s2','s3'],
    ['s1','s2','s3'],
    ['a1','a2','a3'],
    ['a1','a2','a3'],
    ['i1','i2','i3'],
], wspace=0.3)

def run(j):
    I = [0]
    for i in range(time-1):
        I.append(abs(I[-1]+np.random.normal(.1, 3)))
    population_behaviour(LIFPopulation, I, p, postfix=str(j),
        w=norm_weights,
        connectivity=internal_rfnopp_connectivity,
        wmean=50.,
        wstd=5.,
        connections_rate=.1)

for i in range(1,4):
    run(i)
p.show()
```



همانطور که مشاهده می‌شود و همانطور که در بررسی رفتار تک نورون دیدیم، فعالیت جمعیت نورونی رابطه‌ی مستقیم با جریان ورودی دارد؛ هرچه جریان ورودی قوی‌تر باشد، فعالیت جمعیت نیز بیشتر خواهد شد. نکته قابل توجه در نمودارهای بالا آن است که نمودار فعالیت جامعه همواره از ۱۰۰ شروع می‌شود. باید دقت شود که نقطه شروع این نمودار، لحظه صفر نیست. نکته آن است که با توجه به یکسان بودن ویژگی‌های ساختاری تمام نورون‌های جامعه (که می‌توانست اینگونه نباشد)، قبل از اولین فعالیت جامعه، همه نورون‌ها شاهد جریان ورودی یکسان و در نتیجه اختلاف پتانسیل یکسان هستند و در یک لحظه خاص، همگی با هم فعال می‌شوند. از آن پس به دلیل وجود تفاوت در اتصالات و وزن اتصالات، همگامی نورون‌ها از بین رفته و شبکه حالت تصادفی پیدا می‌کند. همچنین است برای زمانی که مدتی هیچ نورونی فعالیت نمی‌کند.

2. نوع و تعداد اتصالات

rfcpc: random fixed coupling probab connectivity

rfnopp: random fixed number of presynaptic partners connectivity

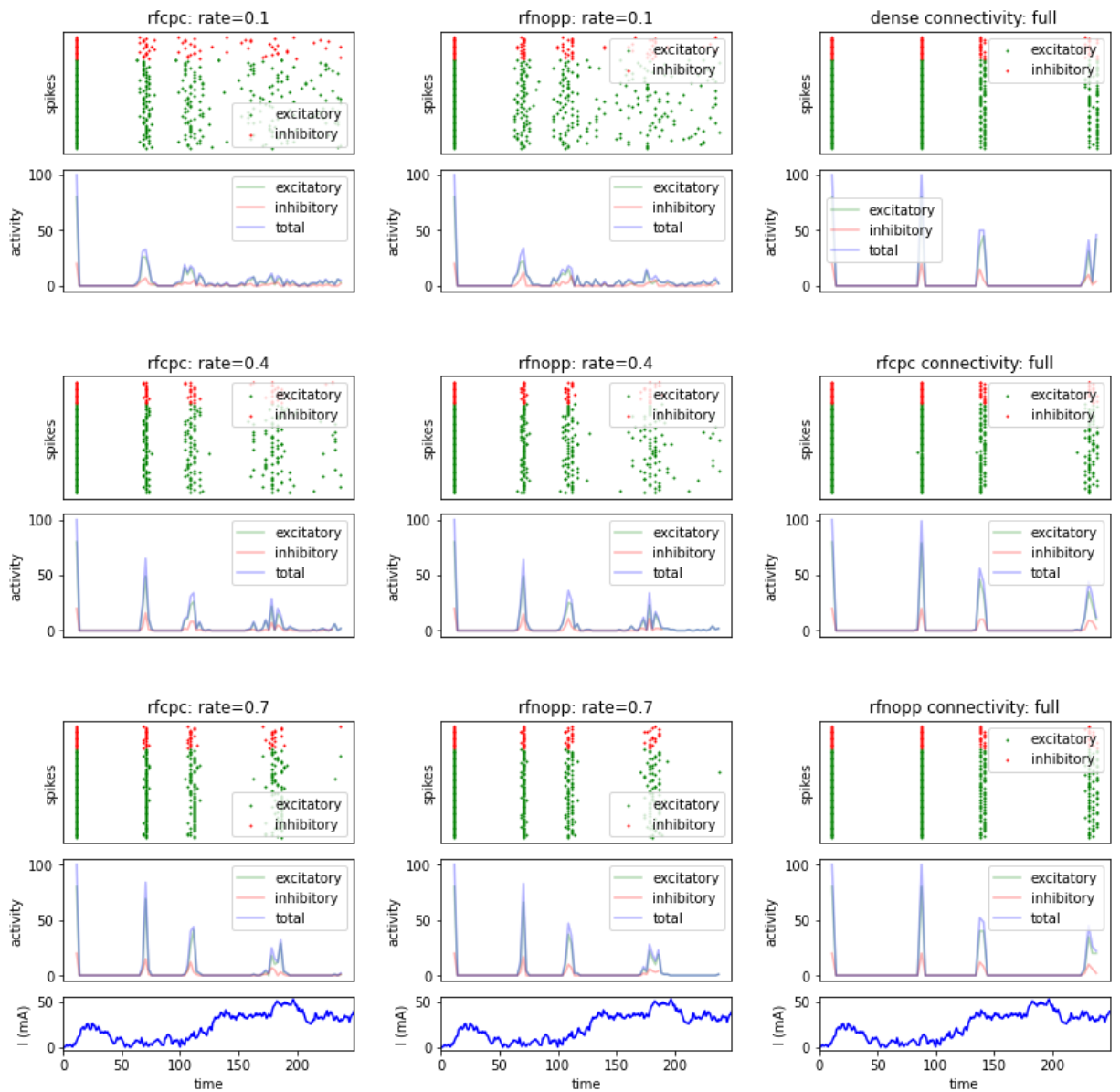
```
In [24]: plt.figure(figsize=(14,14))
p = Plotter([
    ['s11','s21','s31'],
    ['s11','s21','s31'],
    ['a11','a21','a31'],
    ['a11','a21','a31'],
    [None,None,None],
    ['s12','s22','s32'],
    ['s12','s22','s32'],
    ['a12','a22','a32'],
    ['a12','a22','a32'],
    [None,None,None],
    ['s13','s23','s33'],
    ['s13','s23','s33'],
    ['a13','a23','a33'],
    ['a13','a23','a33'],
    ['i13','i23','i33'],
], wspace=0.3, hspace=0.3)
I = [0]
for i in range(time-1):
    I.append(abs(I[-1]+np.random.normal(.1, 3)))

for i,(k,v) in enumerate({"dense connectivity":internal_dense_connectivity,
    "rfcpc connectivity":internal_rfcpc_connectivity,
    "rfnopp connectivity":internal_rfnopp_connectivity}):
    population_behaviour(LIFPopulation, I, p, postfix='3'+str(i+1), name=k+":",
        w=norm_weights,
        wmean=20.,
        wstd=5.,
        connectivity=v,
        connections_rate=1)

for i,r in enumerate([.1,.4,.7]):
    population_behaviour(LIFPopulation, I, p, postfix='1'+str(i+1), name=f"rfcpc {r}")
    w=norm_weights,
    wmean=20.,
    wstd=5.,
    connectivity=internal_rfcpc_connectivity,
    connections_rate=r)

for i,r in enumerate([.1,.4,.7]):
    population_behaviour(LIFPopulation, I, p, postfix='2'+str(i+1), name=f"rfnopp {r}")
    w=norm_weights,
    wmean=20.,
    wstd=5.,
    connectivity=internal_rfnopp_connectivity,
    connections_rate=r)

p.show()
```



مشخص است که حالت dense و حالات اتصالات ۱۰۰ درصدی تفاوتی ندارند و نباید داشته باشند. تفاوت جزئی موجود به دلیل محاسبه تصادفی وزنهای اتصالات می باشد. مشاهده می شود که دو نوع اتصال rfcp و rfnp تفاوت کمی دارند. دلیل این مسئله تعداد زیاد نوروهای جمعیت است که باعث می شود، مطابق با اصل حد مرکزی، این دو نوع اتصال به وضعیت مشابهی برسند. با اینحال تفاوت جزئی وجود دارد و مشاهده می شود که در اتصال rfnp زمان اسپایکها پخش تر است. همچنین مشاهده می شود که با افزایش درصد برقراری اتصالات، نوروها همگام تر شده و همزمان اسپایک می زنند. دلیل آن است که با افزایش اتصالات، تأثیر جریان ورودی به شکل منظم تری در شبکه پخش شده و در نتیجه، نوروها شرایط مشابهی را تجربه می کنند.

3. مقداردهی وزنهای اتصالات

In [32]:

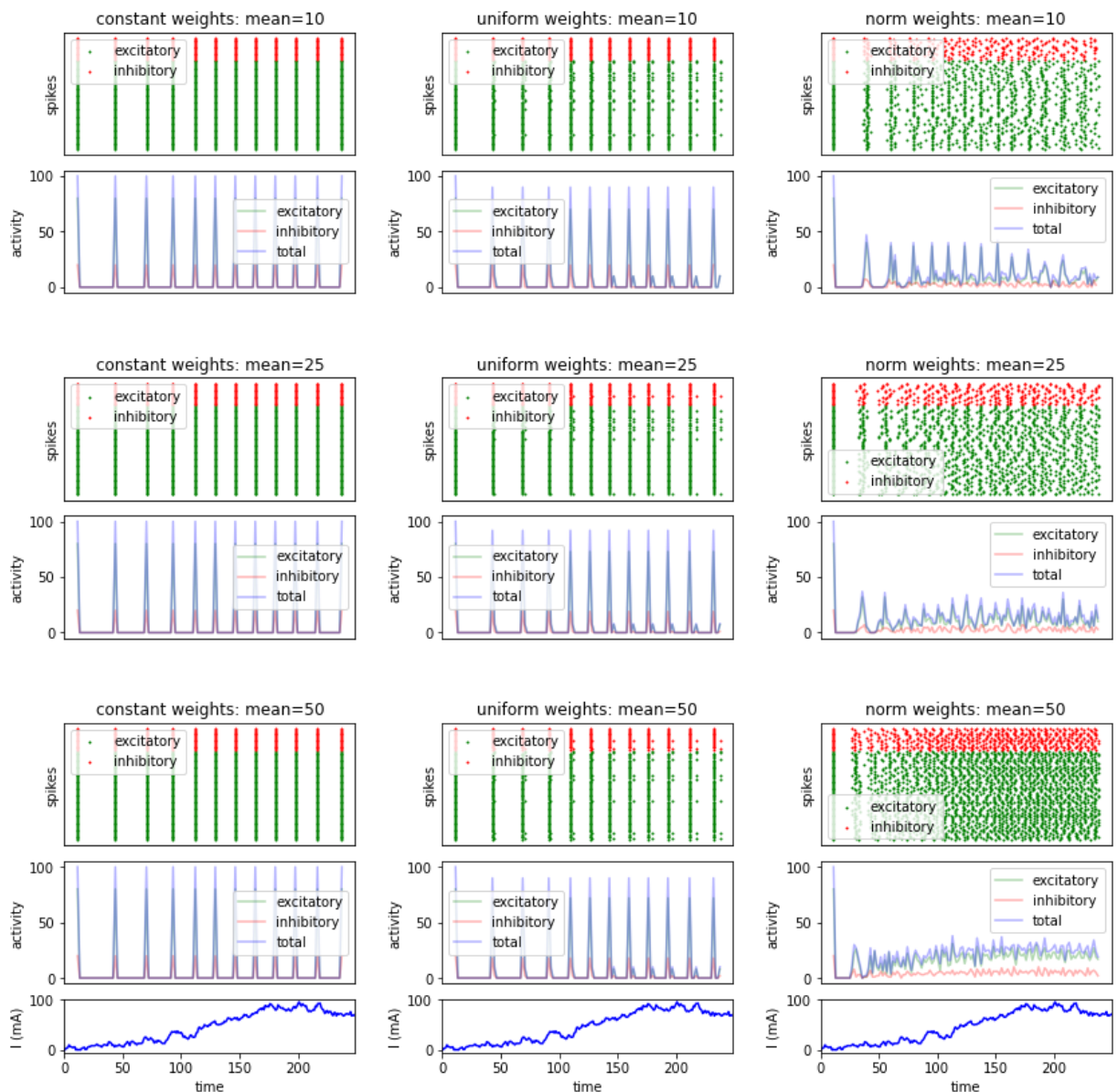
```
plt.figure(figsize=(14,14))
p = Plotter([
    ['s11','s21','s31'],
    ['s11','s21','s31'],
    ['a11','a21','a31'],
    ['a11','a21','a31'],
    [None,None,None],
    ['s12','s22','s32'],
    ['s12','s22','s32'],
    ['a12','a22','a32'],
```

```

['a12','a22','a32'],
[None, None, None],
['s13','s23','s33'],
['s13','s23','s33'],
['a13','a23','a33'],
['a13','a23','a33'],
['i13','i23','i33'],
], wspace=0.3, hspace=0.3)
I = [0]
for i in range(time-1):
    I.append(abs(I[-1]+np.random.normal(.1, 3)))

for i,(k,v) in enumerate({"constant weights":constant_weights,
                           "uniform weights":uniform_weights,
                           "norm weights":norm_weights}.items()):
    for j,mean in enumerate([10,25,50]):
        population_behaviour(LIFPopulation, I, p, postfix=str(i+1)+str(j+1),
                              name=k+": mean="+str(mean),
                              cd=(j==2),
                              w=v,
                              wmean=mean,
                              wstd=10,
                              connectivity=internal_rfnopp_connectivity,
                              connections_rate=.2)
p.show()

```



در نمودارهای بالا، نتایج زیر قابل رویت است:

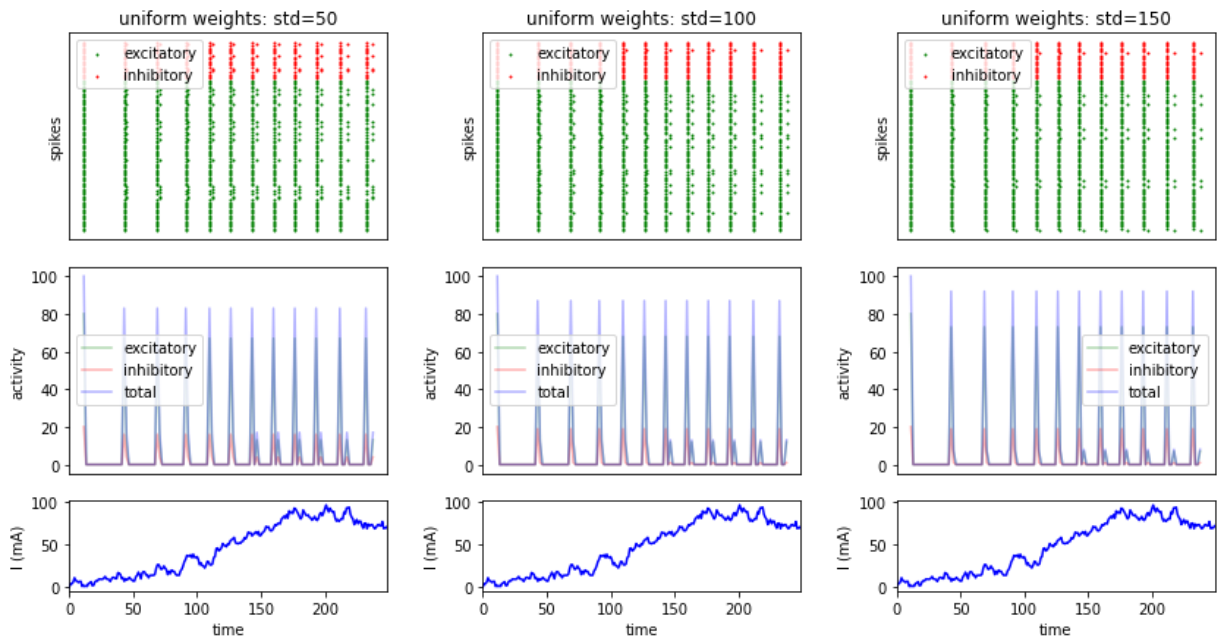
- با توزیع یکنواخت، فارغ از اندازه وزن‌ها، نورون‌های جامعه همگی هماهنگ عمل می‌کنند چرا که همه ورودی دقیقاً یکسانی دریافت می‌کنند (همه نورون‌ها جریان ورودی یکسانی را با وزن یکسانی جابجا می‌کنند).

- در توزیع‌های تصادفی، با افزایش اندازه وزن‌ها، تأثیر برهمکنش‌های نورون‌ها بیشتر شده و در نتیجه از نظم جامعه کاسته می‌شود.

- توزیع نرمال بی‌نظمی بیشتری ایجاد می‌کند چرا که ورودی نورون‌های مختلف تفاوت زیادی را شاهد می‌رود با افزایش واریانس شاهد پراکندگی بیشتر باشیم. این مسئله را بررسی می‌کنیم:

```
In [38]: plt.figure(figsize=(14,7))
p = Plotter([
    ['s1','s2','s3'],
    ['s1','s2','s3'],
    ['a1','a2','a3'],
    ['a1','a2','a3'],
    ['i1','i2','i3'],
], wspace=0.3, hspace=0.3)

for j,std in enumerate([50,100,150]):
    population_behaviour(LIFPopulation, I, p, postfix=str(j+1),
        name="uniform weights: std="+str(std),
        w=uniform_weights,
        wmean=200,
        std=std,
        connectivity=internal_rfnopp_connectivity,
        connections_rate=.2)
p.show()
```



همانطور که شاهد هستیم، بی‌تأثیر است. دلیل این امر آن است که میانگین جریان ورودی به هر نورون تقریباً ثابت است و با توجه به تعداد زیاد نورون‌های جامعه، این تقریب به برابری میل می‌کند.

4. تغییرات ثابت زمانی τ_s

```
In [44]: plt.figure(figsize=(14,7))
p = Plotter([
```

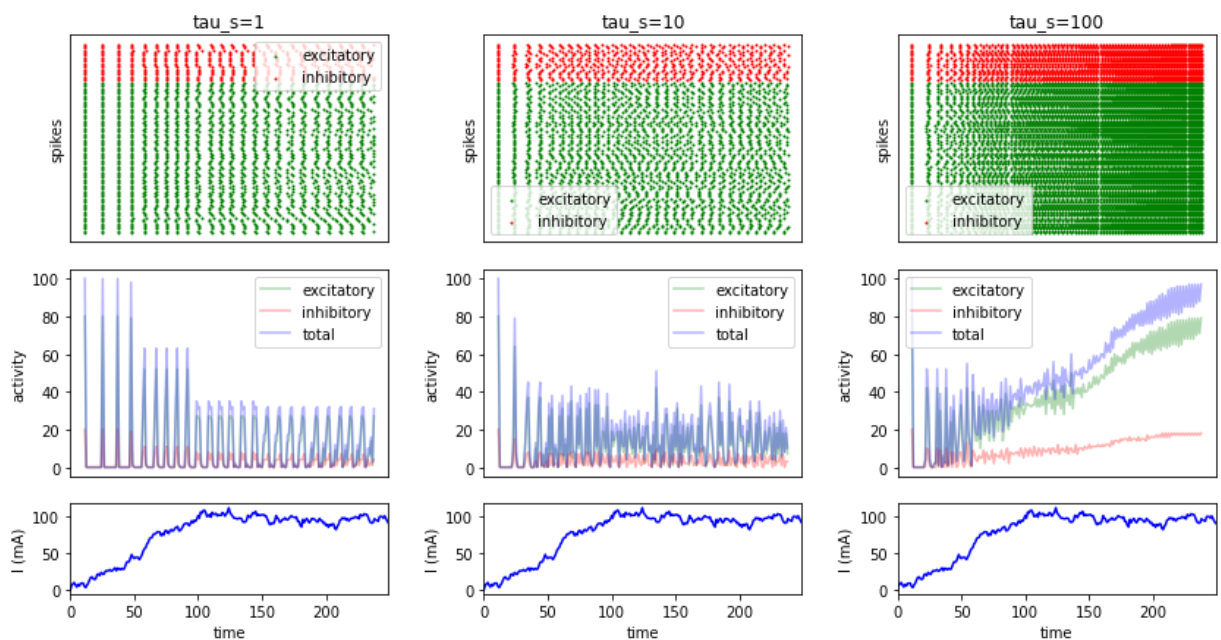
```

['s1','s2','s3'],
['s1','s2','s3'],
['a1','a2','a3'],
['a1','a2','a3'],
['i1','i2','i3'],
], wspace=0.3, hspace=0.3)
I = [0]
for i in range(time-1):
    I.append(abs(I[-1]+np.random.normal(.1, 3)))

for j,tau_s in enumerate([1,10,100]):
    population_behaviour(LIFPopulation, I, p, postfix=str(j+1),
        name="tau_s="+str(tau_s),
        tau_s=tau_s,
        w=norm_weights,
        wmean=25,
        std=5,
        connectivity=internal_rfnopp_connectivity,
        connections_rate=.2)

p.show()

```



به سادگی می‌توان مشاهده کرد که با افزایش ثابت زمانی، فعالیت جامعه افزایش پیدا می‌کند. این مسئله مورد انتظار است چرا که افزایش ثابت زمانی به معنای افزایش ماندگاری اثر اسپایک‌های قدیمی است که باعث می‌شود برهمکنش نوروها اثر بیشتری داشته باشد و فعالیت جامعه افزایش پیدا کند (توجه کنید که تعداد نوروهای تحریکی بیشتر است).